2.1.2.5 Core Research

Over the past year we have supported several core research activities aimed at developing information resources, basic AI research, and tools of general interest to the SUMEX-AIM community. Specific areas of current effort include:

1) The AI Handbook which is a compendium of knowledge about the field of Artificial Intelligence being compiled by students and investigators at several research facilities across the nation. The handbook is broad in scope, covering all of the important ideas, techniques, and systems developed during 20 years of research in AI in a series of articles. Each is about four pages long and is a description written for non-AI specialists and students of AI. The AI Handbook effort is described in more detail in Section 4.2.1 on page 130 and an outline of the current contents of the handbook can be found in Appendix I.

2) The AGE project which is an attempt to isolate inference, control, and representation techniques from previously developed knowledge-based programs; reprogram them for domain independence; write a rule-based interface that will help a user understand what the package offers and how to use the modules; and make the package available to other members of the AIM community. A more detailed description of progress on the AGE package can be found in Section 4.2.2 on page 133.

3) The MAINSAIL project which is an attempt to demonstrate the design of an ALGOL-like language system which facilitates software transportability between different machine/operating system environments. A final report on this effort is given below.

It should be noted that SUMEX is providing only partial support for the AI Handbook and the AGE projects with complementary support coming from an ARPA contract to the Heuristic Programming Project.

MAINSAIL System for Software Transportability

At the end of this grant year the MAINSAIL project will have successfully designed, demonstrated, and documented an ALGOL-like language system for machine-independent software design. This system includes the compiler, code generators, and run-time support for a range of target machine environments including TENEX, TOPS-20, TOPS-10, RT-11, RSX-11, and UNIX. The designs for other environments have been studied but resources have not allowed more extensive implementations. Within Council-approved funding and manpower limits and the AI charter of the SUMEX resource, we do not have access to the more extensive resources that would be required to continue effective development and export of this system beyond this initial research and demonstration phase. We are hopeful that the principal individuals involved (Messrs. Wilcox and Jirak and Ms. Dageforde) will be successful in forming a small private company to support and continue development of MAINSAIL with independent funding from a growing group of potential users. The following is a final summary on this demonstration phase of the MAINSAIL effort. A detailed final report is in preparation.

The primary effort during the past year has been directed at making MAINSAIL a stable, maintainable, complete system ready for distribution and

serious production programming. Implementations developed in prior years have been improved and new ones added. The number of evaluating users has increased, as well as the number of applications programs written in MAINSAIL. The project is now at the point where new implementations can be undertaken, and the groundwork for portability which has been laid over the previous years can now begin to really show its strength.

The compiler has undergone major examination, improvement, and reduction in size of data structures, and as a result is now able to run on machines with small address spaces (e.g., 32K words).

The language itself has remained stable. The runtime system has not undergone any major modifications since September.

Distribution of MAINSAIL beyond its initial test sites has begun. Due to the increasing size of the MAINSAIL user community the need for user support has also increased significantly. As part of our effort to evaluate MAINSAIL's effectiveness in actual applications we have provided user consultation within available resources but we have been limited in the amount of help we could actually provide while continuing active development efforts.

A research project based on MAINSAIL is underway, aimed at providing an efficient program execution and development environment on a high-level language "MAINSAIL machine" which directly executes a tailor-made MAINSAIL instruction set.

a) Implementations

The PDP-10 TENEX version of MAINSAIL has now been in use for about three years at two local sites. A version for a somewhat non-standard TOPS-10 has been used locally to a lesser extent for two years. Standard TOPS-10 was implemented about a year and a half ago and received a moderate amount of use at a remote site. This year standard TOPS-10 was sent to four new sites, including the NIH DCRT Computer Facility.

A TOPS-20 implementation was derived from the TENEX implementation during the past year. The TOPS-20 version is not yet complete in that it is simply a TENEX implementation with a few minor modifications. Utilization of features of the KL-processor instruction set and proper handling of structures in file names have not yet been implemented, but are relatively straightforward additions. This version of the TOPS-20 implementation is now undergoing evaluation at a number of sites, and is beginning to be the most requested version of MAINSAIL.

Due to the interest in using MAINSAIL on machines with small address spaces, substantial development work was done during this past year on PDP-11 implementations.

On many minicomputer configurations the limited address space can have an adverse effect on the performance of large programs. Whenever a working set of modules cannot be contained in primary memory the system begins to exhibit the classic thrashing condition. Since modules are normally swapped from disk the attendant I/O overhead seriously degrades the program's performance.

Some minicomputers have additional memory which is not directly addressable. Typically, this memory can be accessed only by changing a hardware relocation device. A portable caching algorithm has been developed to allow MAINSAIL to take advantage of such memory to reduce the effects of thrashing. The additional memory is used in a two-tier storage hierarchy with the disk. Since access to the additional memory is much faster than access to the disk, swapping from the additional memory takes less time than swapping from the disk. MAINSAIL modules are maintained in the memory cache in a most-recently-used fashion. When the cache fills up, the least recently used module is bumped from the memory cache. The modular design of MAINSAIL made this caching approach quite natural, and holds much promise for further utilization of memory hierarchies.

A PDP-11/40 running the RT-11 operating system has been running MAINSAIL programs for two years. The RSX-11M operating-system interface is complete. A PDP-11/34 running RSX-11M was used as our main testing site during the development of the compiler running in a small address space.

The operating-system interface for UNIX is also complete. A few MAINSAIL utility programs have been run on a PDP-11/34 using UNIX and there are no outstanding problems. This implementation requires further testing.

The runtime system has been run on a standard PDP-11/03 with DEC floppy disks. This was purely a demonstration effort for two main reasons: 1) the floppies are extremely slow, and 2) their storage capacity is insufficient for holding anything other than simple programs, since most of the storage is taken up by the operating system, its utilities, and the MAINSAIL runtimes.

The runtime system and the compiler have been used on a number of LSI-11 configurations. These configurations had either dual density non-DEC floppy disks or an RK equivalent hard disk pack. Some of these machines had an additional 32K words of video memory which MAINSAIL utilized as a module cache. Prior to the demonstration of the compiler under RSX-11M, the fastest PDP-11 compilation on record occurred on an LSI-11 with video memory and an RK type disk.

The operating system interfaces, once written, have caused few problems. There have been two major sources of difficulty in implementing for the PDP-11:

1) The porting of data between machines is often difficult. We are hampered by the availability of compatible peripherals. For instance, our primary RT-11 development machine has no magnetic tape nor floppy disks. It can communicate with other PDP-11's only by exchanging RK-type disk packs or with SUMEX over a 2400 baud terminal line.

2) PDP-11 code generation is non-trivial, and a number of bugs were discovered. This can be contrasted with the PDP-10 code generators, which have caused almost no problems because of the richness of the PDP-10 instruction set, and the ample word size.

The PDP-11 code generation problems have been decreasing in frequency. The demonstration of the compiler on the PDP-11 has increased confidence in the code generators though floating point code generation is only now beginning to undergo extensive testing.

The problems of data exchange have proven substantial. It is difficult to formulate a general strategy due to the diverse file systems which are encountered. These problems are often underestimated, resulting in unexpected delays in the development of portable systems.

A number of groups are interested in the development of other MAINSAIL implementations, including ones for the VAX, ECLIPSE and TI-990 computers. We hope to start work on these implementations this summer through the private company being formed.

b) Distribution and Use

The distribution beyond Stanford of PDP-10 versions of MAINSAIL was begun this year. MAINSAIL has been implemented at sites on the Arpanet, and ported via magnetic tape to other locations. All sites have been able to run MAINSAIL as soon as the files are taken off the tape. This is in contrast to the typical hardware manufacturer's software, which often takes days, or even weeks, to make executable.

There are currently three sites using the TENEX version, six using the TOPS-10 version, and five using the TOPS-20 version.

Extensive work was done this year on various PDP-11 configurations, and it is now beginning to be exported beyond the test sites here at Stanford.

One user has been writing MAINSAIL programs for two years and running them on his PDP-11/40. These include a 3-dimensional graphics package, a code optimizer for the DEC VT-11 display processor, a flow rate monitor for a cell-sorter connected to the PDP-11, and machine-independent arbitrary-precision arithmetic routines.

A geophysics group is writing MAINSAIL programs to extract data from ERTS tapes and to then perform a variety of image analyses on the extracted data. Another user is writing a machine-independent interprocessor communications facility.

A sampling of other MAINSAIL programs developed during the past year include a machine-independent tape transfer program; a program which compares two text files and prints out the differences on a per-line basis; a program which forms a new text file from selected pages of existing text files; a "conference" program enabling more than two people at once to carry out an on-line discussion; a "calculator" program; a record i/o package, which, given a pointer to a record, will print out the values of the fields of the record. Work has also begun on a portable text editor.

A number of sites are now evaluating MAINSAIL with the intent of using it for substantial product development. In most cases, the sites are primarily attracted by MAINSAIL's portability, since there has been no other language which previously played this role while at the same time providing a rich programming environment.

c) Compiler Design

A detailed analysis was made of the compiler, its algorithms and use of data structures. The goal was to reduce the size and number of data structures to allow the compiler to fit on machines with small address spaces, without sacrificing too much efficiency.

Once the compiler was able to fit, the next goal was to improve efficiency and reduce compilation time. First, an analysis of the compiler and of the runtime system uncovered some inefficiencies which were corrected.

Next, various compiler configurations were examined. By configuration is meant the various ways in which procedures can be combined into modules. On a machine with a large address space the compiler is most efficient if it consists of a few modules, since that reduces the number of intermodule calls. But on a machine with a small address space, module swapping is necessary, and compilation time is roughly proportional to the number of swaps. We wanted to determine whether a "better" configuration (one which required less swapping) than that of the existing compiler could be found.

A MAINSAIL program was written to simulate compilation on machines with various address spaces. The simulation was driven by exact data, obtained from traces of all procedure calls made during given compilations. A format was devised for easily specifying potential compiler configurations, and the simulator tested their efficiency. The resulting data showed that curves plotting amount of memory versus number of swaps are smoothly exponential. Examination of this data indicates that for 32K machines, another 10K would cut the number of module swaps in half, thus greatly increasing compilation speed.

As a result, two configurations are now in use: a "big" configuration to be run on machines with "large" address spaces, and a "small" configuration to be run on those with small address spaces. As predicted by the simulation, use of the "optimal" small configuration significantly increased the compilation speed on the PDP-11. Use of the "big" configuration, with just a few large modules, also improved the compiler speed on the PDP-10.

A new approach to code generation has been introduced over the past year. It utilizes tree structures for the intermediate representation, rather than the more primitive triples or quadruples. A tree structure is built for each procedure, and code is generated by walking the tree. This new approach will probably be used in all future code generators since it allows for procedure-wide optimization, and also supports the debugging version described later.

d) Language Design

The language itself has been very stable this past year, undergoing only a few simple additions. The fact that it has remained stable while supporting the past year of development is convincing evidence that the language has matured to the point of commercial viability.

The ability to access certain fields of the array descriptor for an array was added. These fields tell the name and bounds of the array. Similarly, the name of a file can be accessed via a pointer to the file descriptor.

MAINSAIL originally guaranteed ASCII character codes. Last year, for portability reasons, it was decided that MAINSAIL would no longer specify the exact character set used, but only that minimal assumptions would be made about the character set. A number of system procedures were added to complement the guaranteed character set assumptions.

e) Runtime Design

At the time of the last annual report, a new runtime system, oriented toward execution efficiency and less memory utilization, was under implementation. It has been very stable since its completion in September 1978.

Some examples of further improvements are: 1) the ability to have a map of memory printed, showing the number of pages used for control space, data space, and buffers, 2) tuning to the garbage collection facility, and 3) a new response that can be made to an error message will cause the printing of a table listing the procedure calls that led up to the call to the error message routine.

The concept of a "module library" has also been introduced. The output of each compilation (after assembly) is an executable file. When a program consists of a large number of modules, it quickly becomes inconvenient (if not impossible) to have a separate file for each executable module. "Module libraries", bulk repositories for modules, were designed to solve this problem. A utility module was written to provide the necessary management functions, as were procedures to insert and delete library files from a runtime list of libraries maintained by the MAINSAIL system. The MAINSAIL runtimes themselves, along with the compiler modules, now reside in module libraries.

f) Emulation Research

MAINSAIL is being used as the basis of research into a language-oriented approach to program representation and execution. Such an approach starts with language characteristics, which determine program representation (instruction set) and execution environment, which in turn determine the processor architecture. This is in contrast to the conventional machine-oriented approach in which the instruction set and processor architecture exist independently of the language, and hence dictate the representation and limit the execution environment. As technology provides increasing flexibility in machine design, high-level-language processors provide an alternative to general-purpose machine-language processors.

The MAINSAIL compiler, with its retargetable code generators and large body of machine-independent software, is an ideal basis for this study. A comprehensive study is being made of the static and dynamic characteristics of MAINSAIL programs. Based on this study, a number of language representations are obtained by varying two primary design criteria: the nature of an operand and the encoding of the instruction stream. The resulting representations range from a stream of bit-aligned fields which directly reflect the source language structure, to a sequence of simple instructions with highly-constrained operands.

Code generators, as well as instruction-set interpreters, have been developed for a number of such representations. Machine architectures which provide efficient implementations for these representations are also under

exploration. The goal is to provide an extremely efficient MAINSAIL processor from the standpoint of program execution time as well as program development time. Such a processor should be viewed as a "language processor" rather than a general-purpose processor since it is designed explicitly for the purpose of executing a single high-level language. A language processor can be used either as a stand-alone system which serves a single user, or as a component in a larger system consisting of many language processors (which need not all support the same language) that are assigned to appropriate user programs under control of an executive processor.

A MAINSAIL debugger based on this research is operational, though it has not been released for general use. This debugger involves an interpreter for a MAINSAIL instruction set (called "s-code", for structured code) which so closely captures the structure of MAINSAIL that it can be "decompiled" into what is essentially the source text, including the original variable names. The code generator for s-code utilizes the new tree-structured intermediate code, which is unbiased with regard to the form of the target code. The mode of operation on a conventional computer involves compilation of those modules which are to be debugged into s-code. These s-code modules may be freely mixed with native code modules (e.g., modules compiled into the PDP-10 instruction set). During execution, MAINSAIL automatically determines when an s-code module is to gain control, and at that point gives control to the interpreter.

The interpreter allows execution to progress in a manner which directly reflects the source program. The user can single step and place break points on the source-statement level, display and alter the values of variables, and display the decompiled text being executed. A screen-oriented debugger would involve the cursor moving along the displayed text as it was being executed in single-step mode, with the user moving the cursor to points at which break points are to be displayed, or under variables whose values are to be displayed. The current debugger has been designed to support such an approach, but does not yet support this mode of operation.

Program execution can be made to halt based on a variety of conditions such as entry to a particular module or procedure; execution of a particular statement; or upon execution of a specified number of statements since the start of the program. This latter type of break point allows the user to restart a program which encountered an error, and have it break a specified number of statement executions before the error. Single step operation then allows examination of the execution environment on a statement-by-statement basis up to the point of the error. Whenever the s-code interpreter detects an error (e.g., subscript out of range), it gives control to the debugger, which informs the user what module, procedure and statement caused the error, and displays decompiled text around the statement. The user can then use the full power of the debugger to determine the source of the error.

The entire runtime system and compiler can now be interpreted in this fashion. The "MAINSAIL machine" being designed as part of the research will directly execute the s-code representation, i.e., s-code is the (macro) instruction set of the machine. Due to the compactness of s-code (approximately one-third the size of equivalent PDP-10 code), and its transparency with respect to the MAINSAIL execution environment, the MAINSAIL machine will provide optimized program execution along with the debugging capabilities. Since s-code

is the instruction set of the MAINSAIL machine, all modules can be decompiled and debugged with no penalty in execution speed.

2.1.2.6 User Software and Intra-Community Communication

We have continued to assemble and maintain a broad range of utilities and user support software. These include operational aids, statistics packages, DEC-supplied programs, improvements to the TOPS-10 emulator, text editors, text search programs, file space management programs, graphics support, a batch program execution monitor, text formatting and justification assistance, and magnetic tape conversion aids. Over the past year we have undertaken several significant development efforts to provide needed new programs to the SUMEX-AIM community. These include:

1) TTYFTP - A number of users have had the need to move files between their local machines and SUMEX but were not connected to the ARPANET. These include for example the transfer of data between the PUFF project at Pacific Medical Center in San Francisco and SUMEX, distribution of MAINSAIL to various non-network sites, and movement of instrument data in support of the DENDRAL or Ultrasound Imaging (Ob-Gyn) projects. We have undertaken development of a file transfer program usable over any teletype line (hardline, dial-up, TYMNET, etc.) which incorporates appropriate control protocols and error checking. The design is based on the DIALNET protocols designed by Crispin at the Stanford AI Laboratory. Differences from DIALNET were necessary to achieve machine and data source independence. We also expanded the DIALNET packet opcodes to include a new packet (RCT) which prevents data overruns and augmented the DIALNET "request for connection" packet to contain additional needed parameters. TTYFTP is written in MAINSAIL so that we can take advantage of the machine independence inherent in the language. The program is written modularly, and has a scheduler module which can service up to eight FTP modules per line, one packet processor per line, and multiple lines. Because of this it can run as a either user process, or a server process. The latter can be either a listening server (handling in-coming lines) or a host server, started up by a user program and then logged off after all transfers are complete. We have preserved DIALNET compatibility so that we will be able to communicate to machines running DIALNET. After the TENEX implementation is completed, we will make the changes necessary to connect TENEX to a PDP-11 RT-11 system and follow that with an RSX-11M version. Since MAINSAIL is up and running under all three of these operating systems, this process is greatly simplified.

2) EMACS - We have continued to import and support the EMACS text editing system from MIT. This editor offers a broader range of services than TVEDIT but has lacked a smoothly human engineered interface. Substantial effort has gone into developing macro packages that improve the human engineering features of EMACS and providing introductory documentation for new users. This has been closely coordinated with similar efforts at SRI and MIT. A community of EMACS users is now developing at SUMEX.

3) ARCHED - In order to facilitate management of file archive directories, we have been developing a display-oriented editor to give improved interaction when posting retrieval requests and to allow records of previously archived files to have descriptive comments attached, be expunged (because they are outdated), or be moved into secondary archive directories. Facilities will exist to allow viewing files based on name template specifications or date constraints.

We have also made changes and updates to many of the existing programs. While many of these changes were maintenance bug fixes, major efforts were involved to bring up new versions of PASCAL, SAIL BACKUP, MACRO, LINK10, GLOB, PA1050, and a new set of utility routines used by many of the DEC CUSP's. Improvements were made in PUB (a text formatting program), MSG (a message reading program written by J. Vittal), and BBD (the bulletin board reading program developed at SUMEX). Several other new programs are in various stages of being brought up on the system including Knuth's text publication system, TEX; a program to periodically update a news summary file from the AP news service files kept at the Stanford AI Laboratory, APNEWS; a program to connect to the Stanford Center for Information Processing machines, GOTRAN; an improved program to locate users on the SUMEX system and on other ARPANET sites, FIND; and an improved mail facility for GUESTS.

2.1.2.7 Documentation and Education

We have spent considerable effort to develop, maintain, and facilitate access to our documentation so as to accurately reflect available software. The HELP and Bulletin Board subsystems have been important in this effort. As subsystems are updated, we generally publish a bulletin or small document describing the changes. As more and more changes occur, it becomes harder and harder for users to track down all of the change pointers. We are in the process of reviewing the existing documentation system again for compatibility with the programs now on line and to integrate changes into the main documents. This will also be done with a view toward developing better tools for maintaining up-to-date documentation.

2.1.2.8 Software Compatibility and Sharing

At SUMEX-AIM we firmly believe in importing rather than reinventing software where possible. As noted above, a number of the packages we have brought up are from outside groups. Many avenues exist for sharing between the system staff, various user projects, other facilities, and vendors. The advent of fast and convenient communication facilities coupling communities of computer facilities has made possible effective intergroup cooperation and decentralized maintenance of software packages. The TENEX sites on the ARPANET have been a good model for this kind of exchange based on a functional division of labor and expertise. The other major advantage is that as a by-product of the constant communication about particular software, personal connections between staff members of the various sites develop. These connections serve to pass general information about software tools and to encourage the exchange of ideas among the sites. Certain common problems are now regularly discussed on a multi-site level. We continue to draw significant amounts of system software from other

ARPANET sites, reciprocating with our own local developments. Interactions have included mutual backup support, experience with various hardware configurations, experience with new types of computers and operating systems, designs for local networks, operating system enhancements, utility or language software, and user project collaborations. We have been able to import many new pieces of software and improvements to existing ones in this way. Examples of imported software include the message manipulation program MSG, TENEX SAIL, PASCAL, TENEX SOS, INTERLISP, the RECORD program, ARPANET host tables, and many others. Reciprocally, we have exported our contributions such as the crash analysis program, drum page migration system, KI-10 page table efficiency improvements, GTJFN enhancements, PUB macro files, the bulletin board system, MAINSAIL, SPELL, SNDMSG enhancements, our BATCH monitor, and improved SA-10 software.

We have also assisted groups that have interacted with SUMEX user projects get access to software available in our community. For example, Prof. Dreiding's group in Switzerland became interested in some of the system software available here after attending the DENDRAL CONGEN workshops (see Section 4.2.3 on page 139). We have provided him with the non-licensed programs requested.

2.1.3 Resource Management

2.1.3.1 Organization

The SUMEX-AIM resource is administered between the Departments of Genetics and Computer Science of Stanford University. Its mission, locally and nationally, entails both the recruitment of appropriate research projects interested in medical AI applications and the catalysis of interactions among these groups and the broader medical community. User projects are separately funded and autonomous in their management. They are selected for access to SUMEX on the basis of their scientific and medical merits as well as their commitment to the community goals of SUMEX. Currently active projects span a broad range of application areas such as clinical diagnostic consultation, molecular biochemistry, belief systems modeling, mental function modeling, and instrument data interpretation (descriptions of the individual collaborative projects are in Section 4 beginning on page 64).

At the end of the last grant year, Professor Lederberg assumed his new role as president of Rockefeller University and Professor Feigenbaum, chairman of the Stanford Department of Computer Science, took over as principal investigator of the SUMEX project. This management transition took place without missing a beat and the SUMEX-AIM community continues to function with the same high level of vitality as before. This is due, in large part, to the depth of Professor Feigenbaum's prior involvement as co-principal investigator and Stanford's multi-disciplinary support of SUMEX-AIM. Professor Lederberg continues an active role in the SUMEX-AIM community as chairman of the AIM Executive Committee and on a more frequent basis through the system message facilities. Professor Stanley Cohen has continued his role on the Stanford SUMEX Advisory Committee and has assumed a new role on the national AIM Executive Committee. He provides biomedical ties and coordination with the Stanford Medical School and projects.

2.1.3.2 Management Committees

Since the SUMEX-AIM project is a multilateral undertaking by its very nature, we have created several management committees to assist in administering the various portions of the SUMEX resource. As defined in the SUMEX-AIM management plan adopted at the time the initial resource grant was awarded, the available facility capacity is allocated 40% to Stanford Medical School projects, 40% to national projects, and 20% to common system development and related functions. Within the Stanford aliquot, Prof. Feigenbaum and BRP have established an advisory committee to assist in selecting and allocating resources among projects appropriate to the SUMEX mission. The current membership of this committee is listed in Appendix III.

For the national community, two committees serve complementary functions. An Executive Committee oversees the operations of the resource as related to national users and makes the final decisions on authorizing admission for new projects and revalidating continued access for existing projects. It also establishes policies for resource allocation and approves plans for resource

development and augmentation within the national portion of SUMEX (e.g., hardware upgrades, significant new development projects, etc.). The Executive Committee oversees the planning and implementation of the AIM Workshop series currently implemented under Prof. S. Amarel of Rutgers University and assures coordination with other AIM activities as well. The committee will play a key role in assessing the possible need for additional future AIM community computing resources and in deciding the optimal placement and management of such facilities. The current membership of the Executive committee is listed in Appendix III.

Reporting to the Executive Committee, an Advisory Group represents the interests of medical and computer science research relevant to AIM goals. The Advisory Group serves several functions in advising the Executive Committee; 1) recruiting appropriate medical/computer science projects, 2) reviewing and recommending priorities for allocation of resource capacity to specific projects based on scientific quality and medical relevance, and 3) recommending policies and development goals for the resource. The current Advisory Group membership is given in Appendix III.

These committees have actively functioned in support of the resource. Except for the meetings held during the AIM workshops, the committees have "met" by messages, net-mail, and telephone conference owing to the size of the groups and to save the time and expense of personal travel to meet face to face. The telephone meetings, in conjunction with terminal access to related text materials, have served quite well in accomplishing the agenda business and facilitate greatly the arrangement of meetings. Other solicitations of advice requiring review of sizable written proposals are done by mail.

We will continue to work with the management committees to recruit the additional high quality projects which can be accommodated and to evolve resource allocation policies which appropriately reflect assigned priorities and project needs. We will continue to make information available about the various projects both inside and outside of the community and thereby promote the kinds of exchanges exemplified earlier and made possible by network facilities.

2.1.3.3 New Project Recruiting

The SUMEX-AIM resource has been announced through a variety of media as well as by correspondence, contacts of NIH-BRP with a variety of prospective grantees who use computers, and contacts by our own staff and committee members. The number of formal projects that have been admitted to SUMEX has more than trebled since the start of the project to a current total of 9 national AIM projects and 8 Stanford projects. Others are working tentatively as pilot projects or are under review.

We have prepared a variety of materials for the new user ranging from general information such as is contained in a SUMEX-AIM overview brochure to more detailed information and guidelines for determining whether a user project is appropriate for the SUMEX-AIM resource. Dr. E. Levinthal has prepared a questionnaire to assist users seriously considering applying for access to SUMEX-

AIM. Pilot project categories have been established both within the Stanford and national aliquots of the facility capacity to assist and encourage new projects in formulating possible AIM proposals and pending their application for funding support. Pilot projects are approved for access for limited periods of time after preliminary review by the Stanford or AIM Advisory Group as appropriate to the origin of the project.

These contacts have sometimes done much more than provide support for already formulated programs. For example, Prof. Feigenbaum's group at Stanford previously initiated a major collaborative effort with Dr. Osborn's group at the Institutes of Medical Sciences in San Francisco. This project in "Pulmonary Function Monitoring and Ventilator Management - PUFF/VM" (see Section 4.1.7 on page 98) originated as a pilot request to use MLAB in a small way for modeling. Subsequently the AI potentialities of this domain were recognized by Feigenbaum, Nii, and Osborn and a joint proposal was submitted to and funded by NIH. This past summer John Kunz from Dr. Osborn's laboratory spent approximately half time at Stanford to learn more about AI research and to participate more closely in the development of the PUFF/VM program.

Similarly, Prof. Feigenbaum and Ms. Nii recently spent two days with Profs. Kintsch and Polson at the University of Colorado, introducing them to the newly developed AGE package for use in formulating their program on modeling aspects of human cognition.

The following lists the fully authorized projects currently comprising the SUMEX-AIM community (see Section 4 for more detailed descriptions). The nucleus of five projects that were authorized at the initial funding of the resource in December 1973 are marked by "<i>" and the new projects admitted this past year by "<n>".

National Community -

1) Acquisition of Cognitive Procedures (ACT); Dr. J. Anderson (Carnegie-Mellon University)

2) Chemical Synthesis Project (SECS); Dr. T. Wipke (University of California at Santa Cruz)

<n> 3) Hierarchical Models of Human Cognition; Drs. W. Kintsch and P. Polson (University of Colorado)

<i> 4) Higher Mental Functions Project; K. Colby, M.D. (University of California at Los Angeles)

5) INTERNIST Project; J. Myers, M.D. and Dr. H. Pople (University of Pittsburgh)

6) Medical Information Systems Laboratory (MISL); M. Goldberg, M.D. and Dr. B. McCormick (University of Illinois at Chicago Circle)

7) Pulmonary Function Project (PUFF/VM); J. Osborn, M.D. (Institutes of Medical Sciences, San Francisco) and Dr. E. Feigenbaum (Stanford University)

(i) 8) Rutgers Computers in Biomedicine; Dr. S. Amarel (Rutgers University)

9) Simulation of Comprehension Processes; Drs. J. Greeno and A. Lesgold (University of Pittsburgh)

Stanford Community -

1) AI Handbook Project; Dr. E. Feigenbaum

(i) 2) DENDRAL Project; Drs. C. Djerassi and E. Feigenbaum

3) Generalization of AI Tools (AGE); Dr. E. Feigenbaum

4) Large Multi-processor Arrays (HYDROID); Dr. G. Wiederhold

5) Molecular Genetics Project (MOLGEN); Dr. E. Feigenbaum and L. Kedes, M.D.

(i) 6) MYCIN Project; E. H. Shortliffe, M.D. and Dr. B. Buchanan

(i) 7) Protein Structure Modeling; Drs. E. Feigenbaum and R. Engelmore

(n) 8) RX Project; R. Blum, M.D.

As an additional aid to new projects or collaborators with existing projects, we provide a limited amount of funds for use to support terminals and communications needs of users without access to such equipment. We are currently providing support for 6 terminals and 4 modems for users as well as a leased line between Stanford and the University of California at Santa Cruz for the Chemical Synthesis project.

2.1.3.4 Stanford Community Building

The Stanford community has undertaken several internal efforts to encourage interactions and sharing between the projects centered here. Professor Feigenbaum organized a project with the goal of assembling a handbook of AI concepts, techniques, and current state-of-the-art. This project has had enthusiastic support from the students and substantial progress made in preparing many sections of the handbook (see Section 4.2.1 on page 130 for more details).

Weekly informal lunch meetings (SIGLUNCH) are also held between community members to discuss general AI topics, concerns and progress of individual projects, or system problems as appropriate. In addition, presentations from a substantial number of outside speakers are invited.

2.1.3.5 Existing Project Reviews

We have conducted a continuing careful review of on-going SUMEX-AIM projects to maintain a high scientific quality and relevance to our medical AI goals and to maximize the resources available for newly developing applications projects. At the last AIM workshop, meetings of the AIM Advisory Group and Executive Committee were held to review the national AIM projects. These groups recommended continued access for all formal projects then on the system. They also recommended phasing out the Organ Culture pilot project.

In the fall of 1978, meetings of the Stanford Advisory Group were held to review projects supported out of the Stanford aliquot. The recommendation of this group was to phase out support for the Hydroid Project, pending work more directly applicable to SUMEX-AIM goals. The group also recommended phasing out the Quantum Chemistry and Genetics Applications pilot projects unless stronger AI relevance were established immediately. The Quantum Chemistry project is attempting to develop ties to the DENDRAL stereochemistry effort. Since Prof. Loew will move to Rockefeller University this summer, her access to SUMEX would come under the jurisdiction of the AIM Executive Committee and we will ask them to review her application for continued support. The Genetics Application project has acquired their own machine for statistical calculations on genetic demographic data and has stopped using SUMEX.

2.1.3.6 AIM Workshop Support

The Rutgers Computers in Biomedicine resource (under Dr. Saul Amarel) has organized a series of workshops devoted to a range of topics related to artificial intelligence research, medical needs, and resource sharing policies within NIH. Meetings have been held for the past several summers at Rutgers.

In May 1979, a mini-AIM workshop devoted to clinical diagnosis programs was organized by MIT-Tufts and Rutgers and held in Vermont. This meeting was small (about 25 attendees) and emphasized detailed technical discussions about system designs and the strengths and weaknesses of various approaches. Many of the attendees were graduate students in order to maximize the benefit of personal contacts and discussions for on-going research projects. Topics covered in the discussions included state-of-the-art in explanation, causality in reasoning, strategies of focusing and dealing with multiple diagnostic problems, issues of representation and grain of description, creating and updating a knowledge base, planning strategies, issues of time representation, and inexact reasoning.

The SUMEX facility has served as a communications base for workshop planning and provided support for workshop demonstrations when requested. We expect to continue this support for future workshops. The AIM workshops provide much useful information about the strengths and weaknesses of the performance programs both in terms of criticisms from other AI projects and in terms of the needs of practicing medical people. We plan to continue to use this experience to guide the community building aspects of SUMEX-AIM.

2.1.3.7 Resource Allocation Policies

As the SUMEX facility has become increasingly loaded, a number of diverse and conflicting demands have arisen which require controlled allocation of critical facility resources (file space and central processor time). We have already spelled out a policy for file space management; an allocation of file storage is defined for each authorized project in conjunction with the management committees. This allocation is divided among project members in any way desired by the individual principal investigators. System allocation enforcement is implemented by project each week. As the weekly file dump is done, if the aggregate space in use by a project is over its allocation, files are archived from user directories over allocation until the project is within its allocation.

We have implemented effective system scheduling controls (see page 16) to attempt to maintain the 40:40:20 balance in terms of CPU utilization and to avoid system and user inefficiencies during overload conditions. The initial complement of user projects justifying the SUMEX resource was centered to a large extent at Stanford. Over the past five years of the SUMEX grant, a substantial growth in the number of national projects was realized. During the same time the Stanford group of projects has matured as well and in practice the 40:40 split between Stanford and non-Stanford projects is not ideally realized (see Figure 11 on page 49 and the tables of recent project usage on page 52). Our job scheduling controls bias the allocation of CPU time based on percent time consumed relative to the time allocated over the 40:40:20 community split. The controls are "soft" however in that they do not waste computer cycles if users below their allocated percentages are not on the system to consume the cycles. The operating disparity in CPU use to date reflects a substantial difference in demand between the Stanford community and the developing national projects, rather than inequity of access. For example, the Stanford utilization is spread over a large part of the 24-hour cycle, while national-AIM users tend to be more sensitive to local prime-time constraints. (The 3-hour time zone phase shift across the continent is of substantial help in load balancing.) During peak times under the new overload controls, the Stanford community still experiences mutual contentions and delays while the AIM group has relatively open access to the system. For the present, we propose to continue our policy of "soft" allocation enforcement for the fair split of resource capacity.

Our system also categorizes users in terms of access privileges. These comprise fully authorized users, pilot projects, guests, and network visitors in descending order of system capabilities. We want to encourage bona fide medical and health research people to experiment with the various programs available with a minimum of red tape while not allowing unauthenticated users to bypass the advisory group screening procedures by coming on as guests. So far we have had relatively little abuse compared to what other network sites have experienced, perhaps on account of the personal attention that senior staff gives to the logon records, and to other security measures. However, the experience of most other computer managers behooves us to be cautious about being as wide open as might be preferred for informal service to pilot efforts and demonstrations. We will continue developing this mechanism in conjunction with management committee policy decisions.

We have also encouraged mature projects to apply for their own machine resources in order to preserve the SUMEX-AIM resource for research and

development efforts and to support projects unable to justify their own machines. The DENDRAL project is currently applying for a VAX machine to support their planned development and program export work. This machine would be integrated with the SUMEX resource through the planned local network and would be dedicated to biomolecular structure elucidation problems. At the same time it would give SUMEX resource staff experience with the VAX architecture in anticipation of projected developments within the ARPANET AI community to move toward that machine for INTERLISP support. Other projects may make similar proposals in the near future.

2.1.4 Future Plans

Our plans for the next grant year are a continuation of the work in progress as discussed earlier. Specific goals are outlined below. Objectives for the individual collaborating projects are discussed in their respective reports (see Section 4 beginning on page 64).

1) RESOURCE OPERATIONS

We will continue to make available to the SUMEX-AIM communities an effective, state-of-the-art facility to support the development of medical AI programs and to facilitate collaborations between community members. Goals include:

a) Continue development of the existing KI-TENEX facility to maximize effectiveness for community use. We expect to continue improving system reliability and efficiency, subsystem software, documentation and user help facilities, and communications facilities.

b) Finish procurement of a satellite machine (DEC 2020) and integrate it into the existing SUMEX-AIM facility. This will include developing necessary hardware and software interfaces (Ethernet) and evolving management policies and tools with the AIM Executive Committee to allocate this resource most effectively to meet community needs. This system will also give us experience with the many issues of distributing computing resources among collaborating projects that we expect to face in future years.

c) Recruit new applications and projects to broaden the range of high quality medical AI applications. Several potential user projects are currently pending review and we will explore others that might be suggested by advisory group members or other contacts. We will continue to review existing projects in relation to SUMEX AI goals and capacity and to encourage the development of independent resources to support mature projects.

d) We plan to work closely with other AIM resource nodes, such as the one at Rutgers, to ensure effective community support between the facilities and to take further advantage of expertise in various user groups for system and user software development.

e) We will submit an application for a follow-on renewal term to the current 3-year grant which terminates in July 1981. This application will focus on continued development of artificial intelligence tools and applications central to the needs of medical science and the development of effective computing resources within the SUMEX-AIM community to enable progress towards those goals.

2) TRAINING AND EDUCATION

Within our resources, we will continue to assist new and established user projects in gaining access to SUMEX-AIM facilities. Collaborating projects will provide their own manpower and expertise for the development and dissemination of their AI programs. Goals include:

a) Continue to provide a high standard of system documentation and limited staff assistance for user problems.

b) Allocate funds approved for "collaborative linkages" in cooperation with the AIM Executive Committee to assist collaborating projects to meet their needs for communication and access to the SUMEX-AIM resource.

c) Provide continued support for the AIM workshop activities in the form of demonstration support, participation in workshop discussions, and assistance for potential pilot users in understanding the SUMEX-AIM community.

3) CORE RESEARCH

Next year, no further work is planned on the MAINSAIL project for which highly successful initial design and demonstration phases were completed this past year.

Our core research work will emphasize continued development of tools of general interest to the SUMEX-AIM community, AI information resources, and basic efforts to understand and build knowledge-based "intelligent agent" programs. This work will complement on-going collaborator project developments by providing links to make more general results available to the entire community. We will continue to provide partial funding for selected individuals in the Stanford Heuristic Programming Project for these core research goals with special relevance to SUMEX medical AI applications. This support is an appropriate share, complementing funding from other sources such as ARPA and NSF.

Attention will be focused on a number of areas of research:

a) AI Handbook - complete publication of Volume I and concentrate on the research, draft, and external review process for Volume II.

b) AGE - improve the user interface to the AGE "tool kit" including tutorial and design assistance subsystems. We will also extend the range of tools available including such mechanisms as backward-chained inference, heuristic search, portions of the MOLGEN "units" package, and semantic networks.

c) Representation - design appropriate symbolic structures for modeling knowledge about a problem. Presently this phase is carried out entirely by system builders. Goals are to codify the knowledge used to make such decisions, both as an aid to the system builders and ultimately to enable programs themselves to choose appropriate representations.

d) Reasoning - model the appropriate inference mechanisms for a problem and build systems that incorporate those models.

e) Knowledge acquisition - design of systems that acquire knowledge by communication with human experts.

f) Multiple uses of knowledge - design of systems that use the symbolic representation of the domain knowledge for additional purposes such as consensus building (accommodating conflicting advice from experts whose competence may be equal but whose "styles" vary), tutoring of human students by employing the knowledge base (both the information it contains and the way it is organized), and explanation (constructing a chain of rules which satisfactorily rationalize the system's behavior to an observer.

2.2 Summary of Resource Usage

The following data give an overview of SUMEX-AIM resource usage. There are four subsections containing data respectively for 1) overall system loading, 2) resource use by community, 3) resource use by project, and 4) network use.

2.2.1 Overall System Loading

The following plots display several different aspects of system loading over the life of the project. These include total CPU time delivered per month, the peak number of jobs logged in, and the peak load average. The monthly "peak" value of a given variable is the average of the daily peak values for that variable during the month. Thus, these "peak" values are representative of average monthly loading maxima and do not reflect the largest excursions seen on individual days.

These data show well the continued growth of SUMEX use and the self-limiting saturation effect of system load average, especially after installation of our overload controls early in 1978. Since late 1976, when the dual processor capacity became fully used, the peak daily load average has remained between about 5.5 and 6. This is a measure of the user capacity of our current hardware configuration and the mix of AI programs.

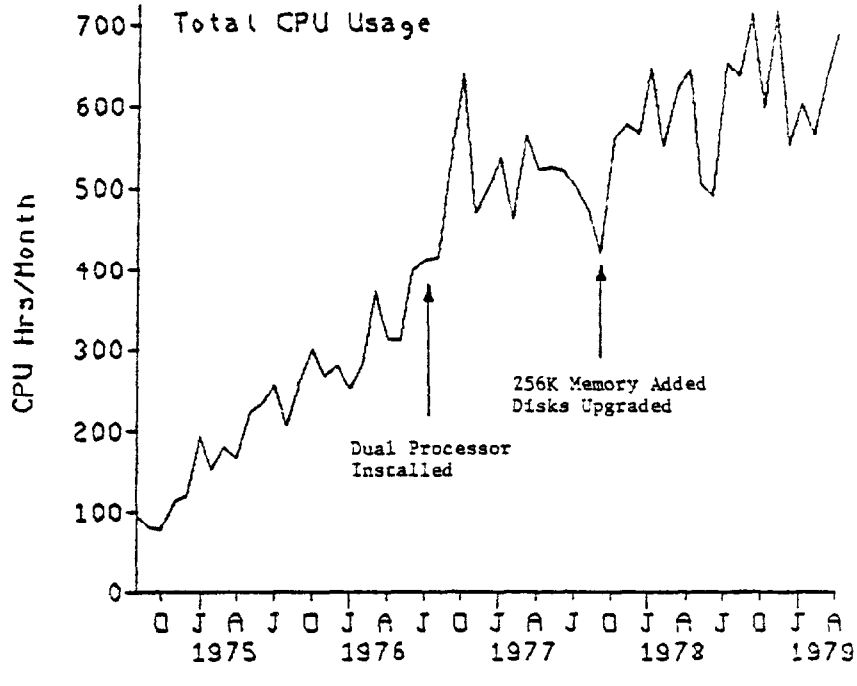

Figure 8. Total CPU Time Consumed by Month

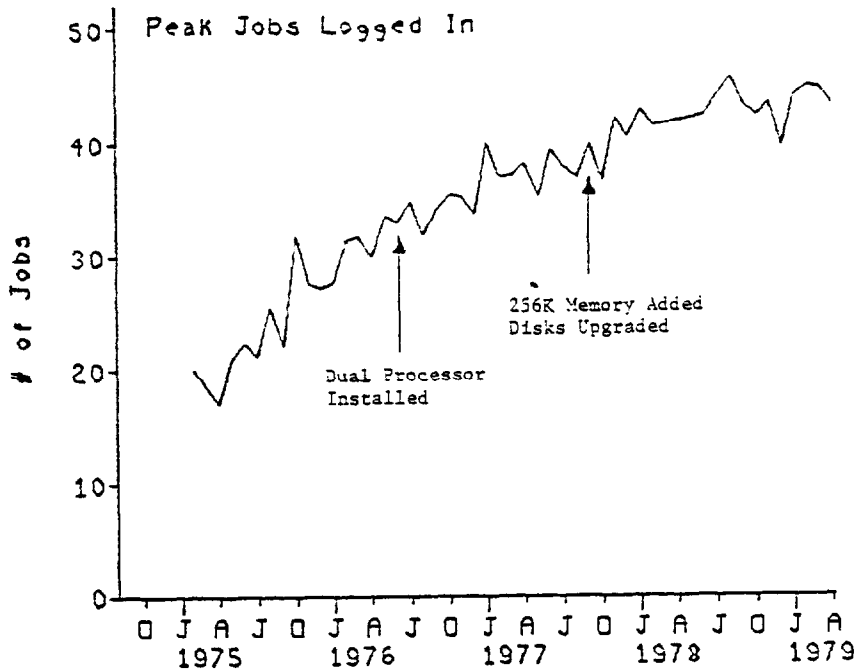

Figure 9. Peak Number of Jobs by Month

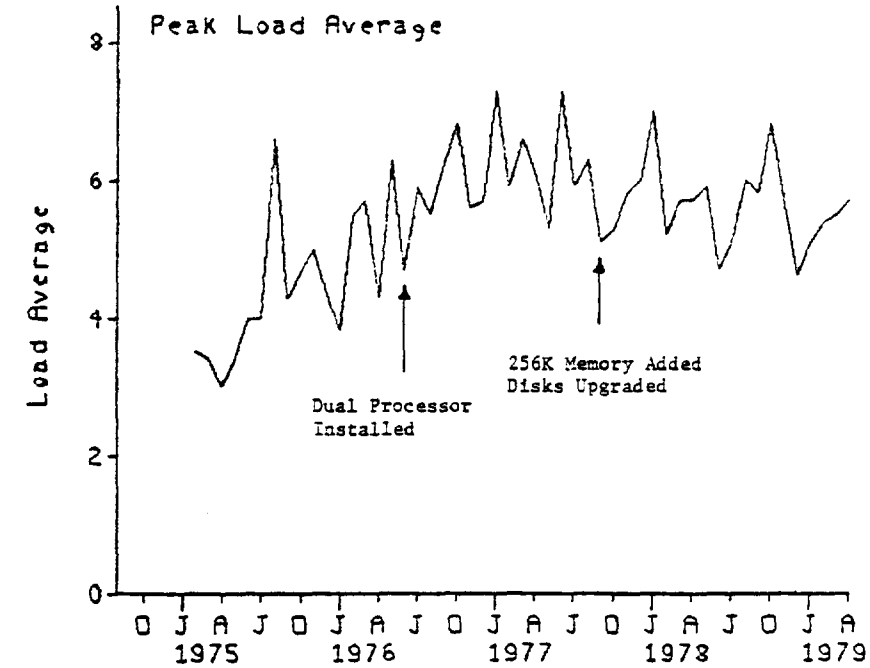

Figure 10. Peak Load Average by Month

2.2.2 Relative System Loading by Community

The SUMEX resource is divided, for administrative purposes, into 3 major communities: user projects based at the Stanford Medical School, user projects based outside of Stanford (national AIM projects), and common system development efforts. As defined in the resource management plan approved by BRP at the start of the project, the available system CPU capacity and file space resources are divided between these communities as follows:

Stanford	40%
AIM	40%
Staff	20%

The "available" resources to be divided up in this way are those remaining after various monitor and community-wide functions are accounted for. These include such things as job scheduling, overhead, network service, file space for subsystems, documentation, etc.

The monthly usage of CPU and file space resources for each of these three communities relative to their respective aliquots is shown in the plots in Figure 11 and Figure 12. Terminal connect time is shown in Figure 13. It is clear that the Stanford projects have held an edge in system usage despite our efforts at resource allocation and the substantial voluntary efforts by the Stanford community to utilize non-prime hours. This reflects the maturity of the Stanford group of projects relative to those getting started on the national side and has correspondingly accounted for much of the progress in AI program development to date.

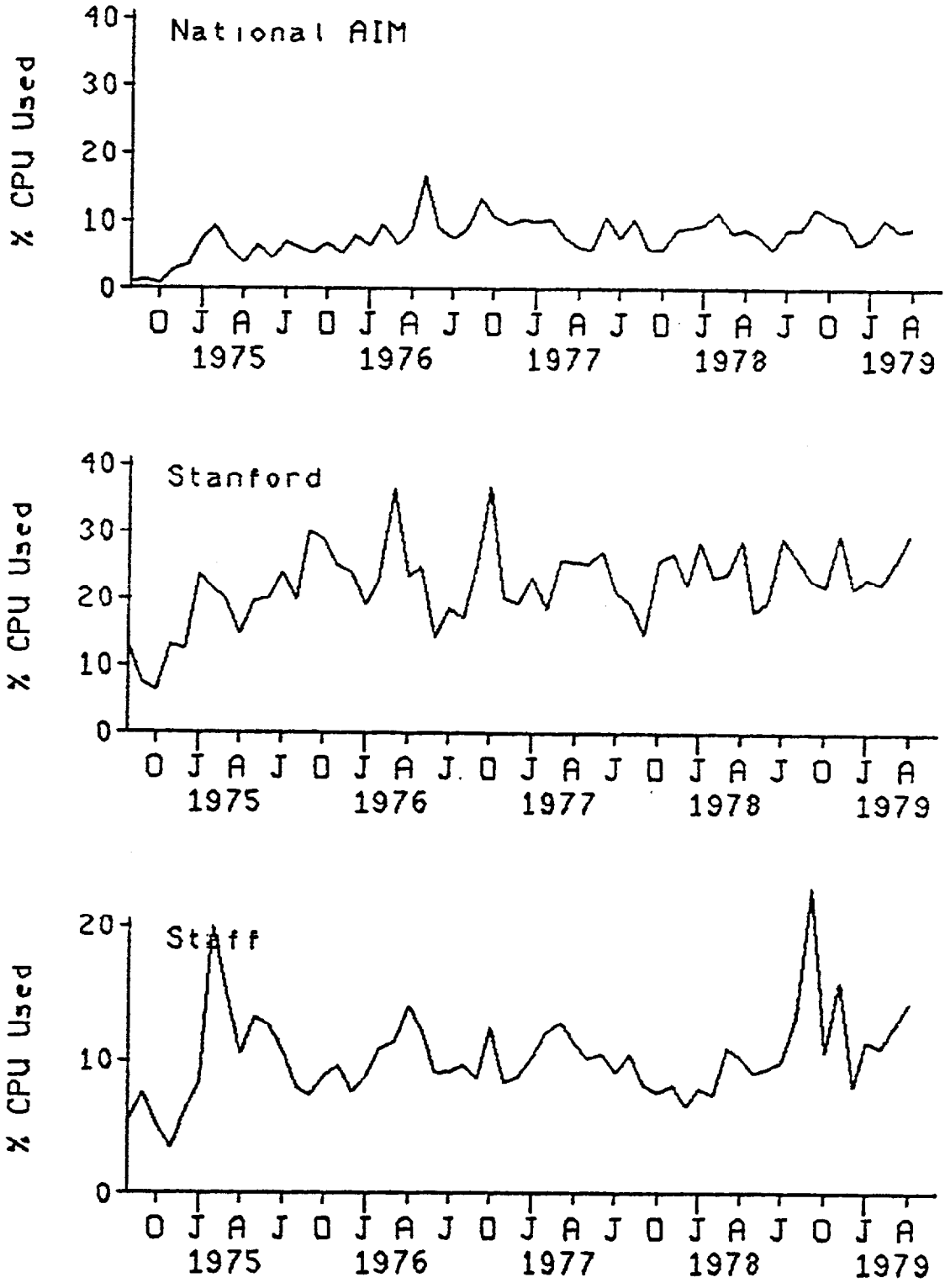

Figure 11. Monthly CPU Usage by Community

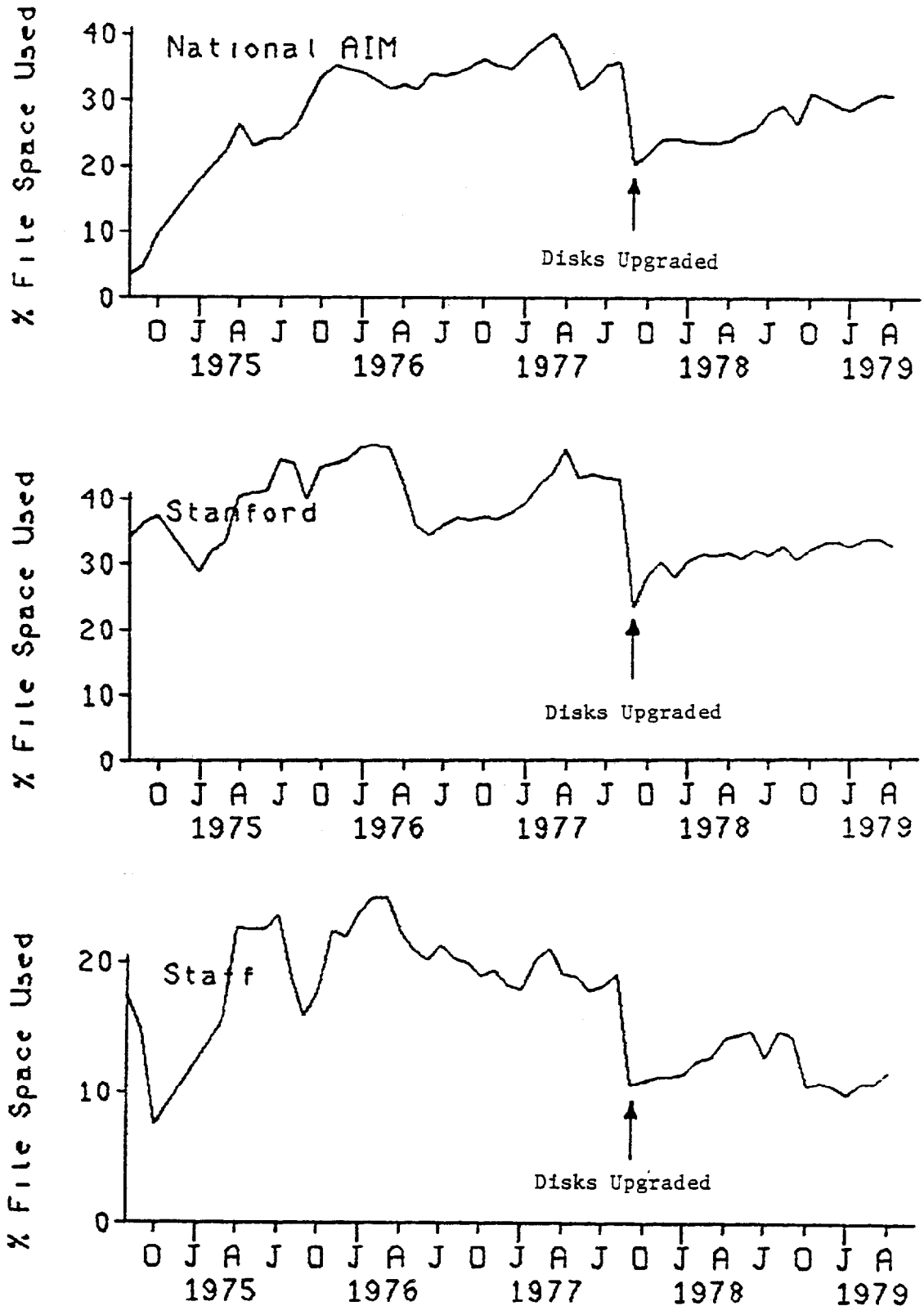

Figure 12. Monthly File Space Usage by Community